mpiBLAST: An Enabling Tool for Interactive Bioinformatics

Moraes, Fernando, Feng, Wu-Chun [*]
Los Alamos National Laboratory, Los Alamos, NM, USA

BLAST [Altschul90, Altschul97] has risen to prominence since its 1990 debut as the application of choice for searching biosequence (genome and protein) databases, and indeed, the most commonly used program in bioinformatics. Due to its speed, roughly seventy percent of bioinformatics clusters run some form of BLAST [Toner02]. Consequently, BLAST is a critical part (as well as a bottleneck) in bioinformatics research. For instance, improved versions of BLAST could have accelerated the identification of SARS and the West Nile Virus.

Because the size of a biosequence database routinely outstrips the memory capacity of a computer, the computer must spend an inordinate amount of time swapping portions of the database to and from disk, which slows a BLAST search by orders of magnitude. Although parallel implementations of BLAST exist, most of them only segment the query among compute nodes. While segmenting the query certainly helps performance, it still does not address the issue of excessive swapping to and from disk.

mpiBLAST addresses the above problem by using an approach based on database segmentation. This approach fragments the database across all the nodes in a cluster computer so that each database fragment fits completely in memory, thus allowing the BLAST search on each node to be done more efficiently. The results from each node are then merged together at the end. Hence, this approach directly solves the problem caused by large databases as well as gaining the benefits of parallel computing. In fact, we have demonstrated superlinear speedup using mpiBLAST, e.g., nearly a 20-fold speed-up with only eight processors.

Since that time, enhancements have been made to both the network I/O subsystem in mpiBLAST and the parallel implementation of the BLAST search algorithm. With respect to the former, the network input/output (I/O) overheads were largely ignored in the past because the original assumption was that the sequence database was being used in a dedicated environment and would therefore not change between consecutive queries. However, such an assumption may not always hold true. Different sequence databases may be required between consecutive queries. Consequently, we must take the distribution of the database into account when executing queries. With a simple but elegant change, that of pipelining the network I/O so as to better distribute the "hot spot" of the shared-database location, we achieve a 35% reduction in network-I/O time. This is of vital importance for the scalability of mpiBLAST; because with every extra node, we put more 'strain' on the shared location.

With respect to the latter, our hybrid database-segmentation/query-segmentation design also shows promise to further extend the scalability of mpiBLAST. Currently we start seeing performance decreases at 128+ nodes, but with this new design we hope to push this 'degradation point' to a much larger number of nodes. The basic goal of this project is to restructure mpiBLAST so as to alleviate current bottlenecks and to provide a better distribution of the work. Currently, a 300-KB query that takes BLAST 22.4 hours on a single processor takes mpiBLAST less than 8 minutes to complete on 128 nodes. Imagine it taking 2 minutes (or less) on 512 workers!

If approved, this poster will feature an overview of past work, detailed performance analysis, and a more detailed glimpse at future improvements that will significantly improve mpiBLAST.